KEY_Lesson18_Dictionaries

December 10, 2021

1 Dictionaries

Today we're going to learn about a super useful Python data type called **dictionaries**. Dictionaries have **key-value pairs** where you can look up a value using a key. Let's jump in with an example.

Here, we're going to make a dictionary of foods where:

- The *keys* are the category
- The *values* are the foods in each category

[1]:
```
# create dictionary
foods = {'vegetable': ['carrot', 'eggplant'], 'fruit': 'cherry'}
# print type of foods
print(type(foods))
# print dictionary
print(foods)
```

```
<class 'dict'>
{'vegetable': ['carrot', 'eggplant'], 'fruit': 'cherry'}
```

Say we want to add an entry to our dictionary. We can do it like this:

[2]:
```
# add entry to dictionary
foods['snack'] = 'chips'
# print dictionary
print(foods)
```

```
{'vegetable': ['carrot', 'eggplant'], 'fruit': 'cherry', 'snack': 'chips'}
```

What if we want to try to get the first element in our foods dictionary?

[3]:
```
# access first element of dictionary
foods[0]
```

```
        ␣
↪---------------------------------------------------------------------------

       KeyError                                  Traceback (most recent call␣
↪last)
```

```
    <ipython-input-3-9635feaa5709> in <module>
      1 # access first element of dictionary
----> 2 foods[0]


    KeyError: 0
```

Whoops! This didn't work. That's because dictionaries are *unordered*. That means that there's no first, second, third etc. element. This is not like Python lists, which are *ordered*. Because of this, we always have to access dictionaries using their keys.

Let's try accessing something in our dictionary using our new knowledge. Let's look up the food(s) from a category (the *value*) using the category (the *key*):

[4]:
```python
# get value in dictionary using key
foods['vegetable']
```

[4]:
```
['carrot', 'eggplant']
```

Cool! As long as we know the category, we can look up the foods that belong to them in our dictionary.

What if we want to print out all the keys or all the values? We can do that like this:

[5]:
```python
# get all the keys:
print(foods.keys())
# get all the values:
print(foods.values())
```

```
dict_keys(['vegetable', 'fruit', 'snack'])
dict_values([['carrot', 'eggplant'], 'cherry', 'chips'])
```

Now let's get a little fancier. Let's try to loop through our dictionary! We're going to use the method `.items()` to access key-value pairs in our dictionary:

[6]:
```python
# print items in dictionary
print(foods.items())
```

```
dict_items([('vegetable', ['carrot', 'eggplant']), ('fruit', 'cherry'),
('snack', 'chips')])
```

[7]:
```python
# loop over dictionary items
for category, food in foods.items():
    print(category)
    print(food)
    print('\n')
```

```
vegetable
['carrot', 'eggplant']
```

```
fruit
cherry


snack
chips
```

Say we decide that we actually want to take 'snack' out of our dictionary for some reason. We can do that using the `del` function:

```
[8]: # delete entry in dictionary and save it to variable
del foods['snack']
# print dictionary
print(foods)
```

```
{'vegetable': ['carrot', 'eggplant'], 'fruit': 'cherry'}
```

Alternatively, if you want to save the values to a variable and remove it from the dictionary at the same time, you can use the `.pop` method:

```
[9]: # delete entry in dictionary and save it to variable
fr = foods.pop('fruit')
# print fr
print(fr)
# print dictionary
print(foods)
```

```
cherry
{'vegetable': ['carrot', 'eggplant']}
```

Nice job! You just learned:

- How to create a dictionary in Python
- How to access keys, values, and key-value pairs in a dictionary
- How to loop over elemets in a dictionary
- How to delete elements from a dictionary